NPS-57He75021

NAVAL POSTGRADUATE SCHOOL

Monterey, California

OPTIMAL SYNTHESIS PROGRAM
FOR
AUTOMATIC CONTROL
(OSPAC)

by

Ronald A. Hess and James W. Sturges

February 1975

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Isham Linder
Superintendent

Jack R. Borsting
Provost

The program described in this report was developed as part of a research effort in pilot-vehicle analysis techniques sponsored by NASA Ames Research Center.

This report was prepared by:

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER NPS-57He75021	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) Optimal Synthesis Program for Automatic Control (OSPAC)	5. TYPE OF REPORT & PERIOD COVERED Technical Report March 19, 74 - February 1975
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ronald A. Hess James W. Sturges	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940 Code 57He	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NASA-DPR-A-98308A
11. CONTROLLING OFFICE NAME AND ADDRESS National Aeronautics and Space Administration Ames Research Center Moffett Field, California 94035	12. REPORT DATE February 1975
	13. NUMBER OF PAGES 57
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Optimal Control
Optimal Estimation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A digital computer program written in Fortran IV is presented which solves the stationary linear quadratic Gaussian optimal control problem. Detailed instructions on the use of the program as well as an illustrative example are presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

I. Background

OSPAC (Optimal Synthesis Program for Automatic Control) is a digital computer program written in Fortran IV, which concerns itself with the stationary linear quadratic Gaussian optimal control problem. This problem can be outlined as follows: Consider a system described by

$$\dot{\underline{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{B}\,\underline{u}(t) + \underline{\gamma}\,\underline{w}(t)$$
$$\underline{y}(t) = \underline{C}\,\underline{x}(t)$$

where

$\underline{A}$	is an n x n plant matrix
$\underline{x}(t)$	is an n x 1 state vector
$\underline{B}$	is an n x p control matrix
$\underline{u}(t)$	is a p x 1 control vector
$\underline{\gamma}$	is an n x t disturbance matrix
$\underline{w}(t)$	is a t x 1 disturbance vector
$\underline{y}(t)$	is a q x 1 output vector
$\underline{C}$	is a q x n output matrix

Here, $\underline{w}(t)$ is a vector of linearly uncorrelated, zero mean white noise signals with Gaussian amplitude probability distribution functions. The elements of $\underline{w}(t)$ are assumed to be sample functions from n random processes which are each ergodic and are jointly ergodic. The covariance matrix for $\underline{w}(t)$ is

$$E\,[\underline{w}(t)\,\underline{w}^T(t+\tau)] = \underline{F}\,\delta(\tau)$$

where $\delta(\tau)$ is the unit impulse function.

The measured quantities or sensor signals are

$$\underline{z}(t) = \underline{H}\,\underline{w}(t) + \underline{v}(t)$$

where

$\underline{z}(t)$ is a u x 1 measurement vector

$\underline{H}$ is a u x n measurement matrix

$\underline{v}(t)$ is a u x 1 measurement noise vector

The elements of $\underline{v}(t)$ are assumed to be sample functions from p random processes each of which are ergodic and jointly ergodic. The covariance matrix for $\underline{v}(t)$ is

$$E\,[\underline{v}(t)\ \underline{v}^T(t + \tau)] = \underline{G}\ \delta(t)$$

The system is assumed to be completely controllable and completely observable. It is desired to find the control function $\underline{u}(t)$ which minimizes the quadratic scalar index of performance

$$J = \lim_{T\to\infty} \frac{1}{T} \int_0^T [\underline{y}^T(t)\ \underline{Q}\ \underline{y}(t) + \underline{u}^T(t)\ \underline{R}\ \underline{u}(t)]\ dt$$

where

$\underline{Q}$ is a q x q symmetric output cost weighting matrix and at least positive semidefinite

$\underline{R}$ is a p x p symmetric control cost weighting matrix and positive definite

The solution to the linear quadratic Gaussian control problem can be outlined as follows:

a.) The optimization problem can, by the called Separation Theorem, be broken up into two separate problems, an optimal control problem and an optimal estimation or filtering problem.

b.) The optimal estimation or filtering problem generates an optimal estimate, $\hat{\underline{x}}(t)$ of the state $\underline{x}(t)$. This estimate is optimal in the sense that

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T \tilde{\underline{x}}^T(t)\ \tilde{\underline{x}}(t)\ dt$$

is minimized, where $\tilde{x}(t)$ is the estimation error defined as

$$\underline{\tilde{x}}(t) = \underline{\hat{x}}(t) - \underline{x}(t)$$

The optimal estimator (or Kalman filter) has the form

$$\dot{\underline{\hat{x}}}(t) = \underline{A}\ \underline{\hat{x}}(t) + \underline{B}\ \underline{u}(t) + \underline{K}\ [\underline{z}(t) - \underline{H}\ \underline{\hat{x}}(t)]$$

The estimator gains are given by

$$\underline{K} = \underline{P}\ \underline{H}^T\ \underline{G}^{-1}$$

where $\underline{P}$ is the error covariance matrix

$$E\ [\underline{\tilde{x}}(t)\ \underline{\tilde{x}}^T(t + \tau)] = \underline{P}\ \delta(t)$$

$\underline{P}$ is the positive definite solution to the steady - state <u>filter matrix Riccati equation</u>

$$\underline{A}\ \underline{P} + \underline{P}\ \underline{A}^T + \underline{\gamma}\ \underline{F}\ \underline{\gamma}^T - \underline{P}\ \underline{H}^T\ \underline{G}^{-1}\ \underline{H}\ \underline{P} = 0$$

c.) The optimal control problem generates an optimal control law $\underline{u}(t)$ which is a linear function of the estimated state

$$\underline{u}(t) = -\ \underline{L}\ \underline{\hat{x}}(t)$$

where $\underline{L}$ is a p x n optimal controller gain matrix. The gain matrix $\underline{L}$ is identical to the one obtained by solving the optimal control problem with no system disturbance, exact state information, and the index of performance given by

$$J = \int_0^\infty [\ \underline{y}^T(t)\ \ \underline{Q}\ \underline{y}(t) + \underline{u}^T(t)\ \ \underline{R}\ \underline{u}(t)]\ dt$$

the controller gain matrix $\underline{L}$ is given by

$$\underline{L} = \underline{R}^{-1}\ \underline{B}^T\ \underline{S}$$

where $\underline{S}$ is the positive definite solution to the steady-state control matrix Riccati equation

$$- \underline{S}\,\underline{A} - \underline{A}^T \underline{S} - \underline{C}^T \underline{Q}\,\underline{C} + \underline{S}\,\underline{B}\,\underline{R}^{-1} \underline{B}^T \underline{S} = 0$$

It can be shown that the state covariance matrix

$$E\,[\underline{x}(t)\,\underline{x}^T(t + \tau)] = (\underline{P} + \underline{M})\,\delta(\tau)$$

where $\underline{P}$ is the solution to the filter matrix Riccati equation and $\underline{M}$ is the positive definite solution to

$$(\underline{A} - \underline{B}\,\underline{L})\,\underline{M} + \underline{M}\,(\underline{A} - \underline{B}\,\underline{L})^T + \underline{K}\,\underline{G}\,\underline{K}^T = 0$$

In addition to the solutions outlined above, it can be shown that the transfer matrix relating the Laplace transform of the optimal control law $\underline{u}(t)$ to the Laplace transform of the measurement vector $\underline{z}(t)$ (with $\underline{v}\,(t) \equiv 0$) is given by

$$\underline{U}\,(S) = - \underline{L}\,(S\underline{I} - \underline{A} + \underline{B}\,\underline{L} + \underline{K}\,\underline{H})^{-1}\,\underline{K}\,\underline{Z}\,(S)$$

where

$$\underline{U}\,(S) = \mathcal{L}\,[\underline{u}\,(t)]$$

$$\underline{Z}\,(S) = \mathcal{L}\,[\underline{z}\,(t)]$$

In addition, the characteristic roots of the estimator are the roots of

$$|\,S\,\underline{I} - (\underline{A} - \underline{K}\,\underline{H})\,| = 0$$

and the characteristic roots of the state-feedback controller are the roots of

$$|\,S\,\underline{I} - (\underline{A} - \underline{B}\,\underline{L})\,| = 0$$

The characteristic roots of the entire closed-loop system, i.e., the plant, estimator and state-feedback controller are just the estimator roots and state feedback controller roots taken together.

II. OSPAC Description

A. Introduction

OSPAC makes extensive use of the Variable Dimension Automatic Synthesis Program (VASP) configured by John S. White and Homer Q. Lee of NASA Ames Research Center. A documentation report entitled "Users Manual for the Variable Dimension Automatic Synthesis Program (VASP)," Oct. 1971, may be obtained from NTIS (N72-10190). OSPAC can provide the following output:

1.) $\underline{S}$, the solution to the steady-state control matrix Riccati equation.

2.) $\underline{L}$, the controller gain matrix.

3.) $\underline{P}$, the solution to the steady-state filter matrix Riccati equation.

4.) $\underline{K}$, the estimator (filter) gain matrix

5.) $\underline{P} + \underline{M}$, the covariance of the system state

6.) CVHX, the covariance of $\underline{H}\ \underline{x}(t)$

7.) CVU, the covariance of $\underline{u}(t)$

8.) J, the value of the index of performance

9.) the roots of

$$| S\ \underline{I} - (\underline{A} - \underline{B}\ \underline{L}) | = 0$$

$$| S\ \underline{I} - (\underline{A} - K\ H) | = 0$$

10.) the elements of the transfer matrix relating $\underline{U}(S)$ to $\underline{Z}(S)$

OSPAC solves the steady-state Riccati equations by integrating the differential Riccati equations until a steady-state solution is reached or when the maximum number of integration steps (as specified by the user, see NCONT(2) below) has been reached.

B. OSPAC Input

As presently configured, the maximum dimensions of the input matrices for OSPAC are

$$n = 10$$
$$p = 10$$
$$q = 10$$
$$t = 9$$
$$u = 9$$

It is imperative that, in his program, the user ensure

$$q < n$$
$$t < n$$
$$u < n$$

If the conditions above are not met, subroutine AUG will produce incorrect results.

The input card arrangement is shown in tabular form on the next page. The description of the items in the table follows.

NSOL This single integer, in format (I1) specifies the number of problems to be run.

NOPT This single integer, in format (I1) specifies the solution option. If

NOPT = 1, one obtains only the state-feedback controller solution ($\underline{L}$, $\underline{S}$).

NOPT = 2, one obtains only the estimator or filter solution ($\underline{K}$, $\underline{P}$).

NOPT = 3, one obtains the controller, estimator and covariance solutions ($\underline{L}$, $\underline{S}$, $\underline{K}$, $\underline{P}$, $\underline{P} + \underline{M}$, $\underline{\text{CVHX}}$, $\underline{\text{CVU}}$, J).

NOPT = 4, one obtains the same solutions as in NOPT = 3 plus the system characteristic roots and transfer matrix.

OSPAC Input Cards

Card Number	Input	Format
1	NSOL	(I1)
2	Title for Problem	(A72)
3	NOPT	(I1)
4 +	Input Matrices (ZZ cards)	
	<u>for NOPT = 1</u>: NCONT, <u>A</u>, <u>B</u>, <u>C</u>, <u>Q</u>, <u>R</u> <u>NOPT = 2</u>: NCONT, <u>A</u>, <u>F</u>, <u>G</u>, <u>GAM</u>, <u>H</u> <u>NOPT = 3</u>: NCONT, <u>A</u>, <u>B</u>, <u>C</u>, <u>Q</u>, <u>R</u>, NCONT, <u>F</u>, <u>G</u>, <u>GAM</u>, <u>H</u>, NCONT <u>NOPT = 4</u> Same as for NOPT = 3	(3I10) for NCONT (A4, 4X, 2I4) for header cards (7E10.5) for matrices
4 + ZZ	Title for Next Problem	(A72)

NCONT This vector of length 3 is input for each Riccati solution (S, P and M) in format (3I10)

NCONT(1) = 1

NCONT(2) = maximum number of integration steps in Riccati solution; NCONT(2) should be ≥ 100.

NCONT(3) = 1

A, B, etc. With the exception of NCONT, each input matrix requires a header card in format (A4, 4X, 2I4). This represents a 4 character title, 4 blank spaces, then the number of rows and columns in the matrix. Each row, beginning on a new card is entered after the header card. Since the program can handle some 10 X 10 matrices and since the input format is (7E10.5), some matrices may require 2 cards per row. However, <u>each matrix row must begin on a new card</u>.

C. OSPAC Output

The following problems may occur in some OSPAC executions.

1.) UNDERFLOW Messages; the VASP programs used in OSPAC frequently generate very small numbers which result in UNDERFLOW error messages. The main program includes an ERRSET subroutine which prevents the messages from being printed each time such an "error" occurs. These underflows <u>do not</u> compromise the solution in any way.

2.) OVERFLOW Messages; If OSPAC generates OVERFLOW error messages which are not attributable to user input errors, then input scaling is necessary. This is discussed in Section III

3.) Failure to converge; Each Riccati solution, $\underline{S}$, $\underline{P}$ and $\underline{M}$, is preceded by a statement indicating the number of iterations required to obtain the solution. If this number equals the value of NCONT(2) for that equation, then the solution has not converged. Assuming that the system is controllable and observable, and that NCONT(2) $\geq$ 100, failure to converge usually means that the integration step size is too large. The VASP routines are supposed to automatically adjust the initial stepsize (set equal to 1.0D+00 in the third argument of the subroutines ETPHI called in the main program) for each problem. Occasionally, however, this automatic procedure fails. The user should then reduce the value of the constant in the ETPHI call statement for the particular equation (controller, filter, or state covariance) which failed to converge, and rerun the job.

The appendix provides a listing of the main program and all the subroutines.

III. Scaling Considerations

For large systems ($n \geq 8$), some scaling of the input matrices is often necessary. When OSPAC produces overflow error messages and no obvious source for these errors can be found, scaling is indicated. A simple scaling procedure that has been used with a good deal of success with OSPAC involves amplitude scaling the system equations as though they were going to be programmed on an analog computer. Again consider the system

$$\dot{\underline{x}}(t) = \underline{A}\,\underline{x}(t) + \underline{B}\,\underline{u}(t) + \underline{\gamma}\,\underline{w}(t)$$

$$\underline{y}(t) = \underline{C}\,\underline{x}(t)$$

$$\underline{z}(t) = \underline{H}\,\underline{x}(t) + \underline{v}(t)$$

$$J = \lim_{T \to \infty} \frac{1}{T} \int_0^T [\underline{y}^T(t)\,\underline{Q}\,\underline{y}(t) + \underline{u}^T(t)\,\underline{R}\,\underline{u}(t)]\,dt$$

Now define the following matrices:

$\underline{X}_M$ is an n x n diagonal matrix whose elements, $x_{m_{ii}}$, are "guestimates" of the maximum values of the state variables $x_i(t)$.

$\underline{U}_M$ is a p x p diagonal matrix whose elements, $u_{M_{ii}}$, are "guestimates" of the the maximum values of the controls $u_i(t)$.

$\underline{Y}_M$ is a q x q diagonal matrix whose elements, $y_{M_{ii}}$, are "guestimates" of the maximum values of the output variables $y_i(t)$.

$\underline{Z}_M$ is a u x u diagonal matrix whose elements, $z_{M_{ii}}$, are "guestimates" of the maximum values of the measurements $z_i(t)$.

Now define

$$\underline{x}_s(t) = \underline{X}_M^{-1}\,\underline{x}(t)$$

$$\underline{u}_s(t) = \underline{U}_M^{-1}\,\underline{u}(t)$$

$$\underline{y}_s(t) = \underline{Y}_M^{-1}\,\underline{y}(t)$$

$$\underline{z}_s(t) = \underline{Z}_M^{-1}\,\underline{z}(t)$$

where the subscript 's' refers to scaled quantities. Rewriting the original system equations using the matrices defined above yields

$$\underline{X}_M\,\dot{\underline{x}}_s(t) = \underline{A}\,\underline{X}_M\,\underline{x}_s(t) + \underline{B}\,\underline{U}_M\,\underline{u}_s(t) + \underline{\gamma}\,\underline{w}(t)$$

$$\underline{Y}_M\,\underline{y}_s(t) = \underline{C}\,\underline{X}_M\,\underline{x}_s(t)$$

$$\underline{Z}_M\,\underline{z}_s(t) = \underline{H}\,\underline{X}_M\,\underline{x}_s(t) + \underline{v}(t)$$

$$J = \lim_{T\to\infty} \frac{1}{T}\int_0^T \{[\underline{Y}_M\,\underline{y}_s(t)]^T\,\underline{Q}\,[\underline{Y}_M\,\underline{y}_s(t)] + [\underline{U}_M\,\underline{u}_s(t)]^T\,\underline{R}\,[\underline{U}_M\,\underline{u}_s(t)]\}\,dt$$

These equations can be written

$$\dot{\underline{x}}_s(t) = \underline{A}_s \underline{x}_s(t) + \underline{B}_s \underline{u}_s(t) + \underline{\Upsilon}_s \underline{w}(t)$$

$$\underline{y}_s(t) = \underline{C}_s \underline{x}_s(t)$$

$$\underline{z}_s(t) = \underline{H}_s \underline{x}_s(t) + \underline{u}_s(t)$$

$$J = \lim_{T \to \infty} \frac{1}{T} \int_0^T [\underline{y}_s^T(t) \underline{Q}_s \underline{y}_s(t) + \underline{u}_s^T(t) \underline{R} \underline{u}_s(t)] \, dt$$

with

$$\underline{F}_s = \underline{F}$$

$$\underline{G}_s = \underline{Z}_M^{-1} \underline{G} Z_M^{-1}$$

where

$$\underline{A}_s = \underline{X}_M^{-1} \underline{A} \underline{X}_M$$

$$\underline{B}_s = \underline{X}_M^{-1} \underline{B} \underline{U}_M$$

$$\underline{C}_s = \underline{Y}_M^{-1} \underline{C} \underline{X}_M$$

$$\underline{\Upsilon}_s = \underline{X}_M^{-1} \underline{\Upsilon}$$

$$\underline{H}_s = \underline{Z}_M^{-1} \underline{H} \underline{X}_M$$

$$\underline{Q}_s = \underline{Y}_M^T \underline{Q} \underline{Y}_M$$

$$\underline{R}_s = \underline{U}_M^T \underline{R} \underline{U}_M$$

The scaled matrices above are then used as inputs to OSPAC. The output of OSPAC can then be unscaled to obtain the solution to original problem. Unscaling the pertinent output quanties is summarized below.

$$\underline{P} = \underline{X_M}\ \underline{P_s}\ \underline{X_M}^T$$

$$\underline{L} = \underline{L_s}\ \underline{X_M}^{-1}$$

$$\underline{P} + \underline{M} = \underline{X_M}\ (\underline{P} + \underline{M})_s\ \underline{X_M}^T$$

$$\underline{K} = \underline{K_s}\ \underline{Y_M}^{-1}$$

$$\begin{aligned} \underline{U}(S) &= \underline{U_M}\ \underline{U_s}(S) \\ &= U_M\ [-\underline{L}_s\ (S\underline{I} - \underline{A_s} + \underline{B_s}\ \underline{L_s} + \underline{K_s}\ \underline{H_s})^{-1}\ \underline{K_s}]\underline{Z_M}^{-1}\ \underline{Z}(S) \end{aligned}$$

It should be emphasized that <u>the eigenvalues of the problem are unaffected by amplitude scaling</u>, i.e. the roots of

$$|\ S\ \underline{I} - (\underline{A} - \underline{B}\ \underline{L})\ | = 0$$

and

$$|\ S\ \underline{I} - (\underline{A_s} - \underline{B}_s\ \underline{L}_s)\ | = 0$$

are <u>identical</u>, as are the roots of

$$|\ S\ \underline{I} - (\underline{A} - \underline{K}\ \underline{H})\ | = 0$$

and

$$|\ S\ \underline{I} - (\underline{A}_s - \underline{K}_s\ \underline{H}_s)\ | = 0$$

as are the roots of

$$|\ S\ \underline{I} - \underline{A} + \underline{B}\ \underline{L} + \underline{K}\ \underline{H}\ | = 0$$

and

$$|\ S\ \underline{I} - \underline{A}_s + \underline{B_s}\ \underline{L}_s + \underline{K}_s\ \underline{H}_s\ | = 0$$

IV. Sample Problem - Helicopter Optimal Control Problem

The longitudinal motion of a helicopter near hover in turbulence can be modeled reasonably well by the following set of differential equations

$$\ddot{\theta} - a_1\dot{\theta} - a_2 (u - u_g) - b\delta = 0$$

$$\dot{u} - a_3\dot{\theta} - a_4 (u - u_g) - g(\theta+\delta) = 0$$

where

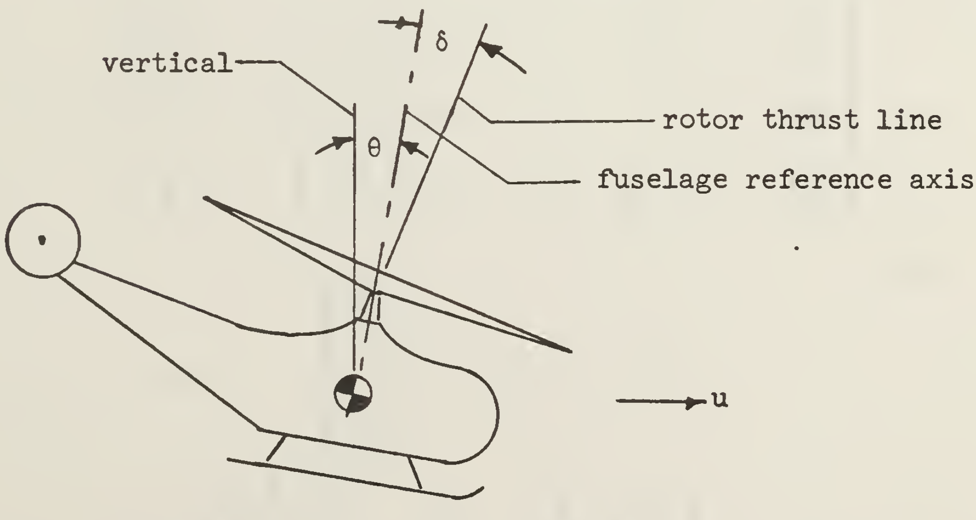
u_g = longitudinal fore-aft turbulence (here assumed to have a white spectrum), ft/sec

θ = pitch angle of fuselage, rad

δ = tilt of rotor tip path plane with respect to fuselage, rad

g = acceleration due to gravity, ft/sec^2

u = groundspeed measured from trim, ft/sec



The synthesis problem centers about finding the control law $\delta(t)$ which minimizes a quadratic index of performance with groundspeed being the measured variable.

$$a_1 = -.4\ /\text{sec} \qquad a_3 = -4.593\ \text{ft/sec}$$

$$a_2 = -.003048/\text{ft-sec} \qquad a_4 = -.02/\text{sec}$$

$$b = -6.3/\text{sec}^2$$

Assume

$$E[u_g(t)\, u_g(t+\tau)] = 25\, \delta(\tau)\ \text{ft}^2/\text{sec}^2$$

The measured variable is u and

$$E[v(t)\, v(t+\tau)] = .01\, \delta(\tau)\ \text{ft}^2/\text{sec}^2$$

One can define a set of state variables

$$x_1 = \theta$$
$$x_2 = \dot{\theta}$$
$$x_3 = u$$

and a set of state equations

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & a_1 & a_2 \\ g & a_3 & a_4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{bmatrix} 0 \\ b \\ g \end{bmatrix} \delta + \begin{bmatrix} 0 \\ -a_2 \\ -a_4 \end{bmatrix} u_g$$

Now

$$z = [0,\ 0,\ 1] \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + v$$

$$\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}$$

$$\underline{Q} = \begin{bmatrix} 100. & 0 & 0 \\ 0 & 100. & 0 \\ 0 & 0 & .04 \end{bmatrix} \quad \underline{R} = 100.$$

Note that, if, at some instant of time

$$\theta = .1 \text{ rad}$$

$$\dot{\theta} = .1 \text{ rad/sec}$$

$$u = 5.0 \text{ ft/sec}$$

$$\delta = .1 \text{ rad}$$

each scalar term in the integrand of the index of performance would be making a contribution of unity to the integrand. This provides some rationale for the selection of the $\underline{Q}$ and $\underline{R}$ matrices above.

The input deck set-up is shown on the next pages. Following that is the OSPAC output. No scaling was necessary in this problem.

Helicopter Optimal Control Problem

Input Deck

1–5	6	7–10	11–20	21–30	31–40
1					
HELIC	Ø	PTER	ØPTIMAL C	ØNTRØL PRØ	BLEM
4					
1			100	1	
A			3 3		
0.			1.	0.	
0.			-.4	-.003048	
32.2			-4.593	-.02	
B			3 1		
0.					
6.3					
32.2					
C			3 3		
1.			0.	0.	
0.			1.	0.	
0.			0.	1.	
Q			3 3		
100.			0.	0.	
0.			100.	0.	
0.			0.	.04	

column

```
R          1   1
100.
1         100       1
F          1   1
25.
G          1   1
.01
GAH        3   1
0.
.003048
.02
H          1   3
0.        0.        1.
1         100       1
```

Output

```
HELICOPTER OPTIMAL CONTROL PROBLEM

          A   MATRIX          3 ROWS          3 COLUMNS
 0.0                   1.0000000D 00     0.0
 0.0                  -4.0000000D-01    -3.0480000D-03
 3.2200000D 01        -4.5930000D 00    -2.0000000D-02

          B   MATRIX          3 ROWS          1 COLUMNS
 0.0
 6.3000000D 00
 3.2200000D 01

          C   MATRIX          3 ROWS          3 COLUMNS
 1.0000000D 00         0.0               0.0
 0.0                   1.0000000D 00     0.0
 0.0                   0.0               1.0000000D 00

          Q   MATRIX          3 ROWS          3 COLUMNS
 1.0000000D 02         0.0               0.0
 0.0                   1.0000000D 02     0.0
 0.0                   0.0               4.0000000D-02

          R   MATRIX          1 ROWS          1 COLUMNS
 1.0000000D 02
             10   ITERATIONS

      S       MATRIX          3 ROWS          3 COLUMNS
 1.9461748D 02         1.2902733D 01     2.4810185D 00
 1.2902733D 01         1.7926885D 01    -2.0154349D-01
 2.4810185D 00        -2.0154349D-01     9.9373886D-02

      L       MATRIX          1 ROWS          3 COLUMNS
 1.6117602D 00         1.0644968D 00     1.9301151D-02

          F   MATRIX          1 ROWS          1 COLUMNS
 2.5000000D 01

          G   MATRIX          1 ROWS          1 COLUMNS
 1.0000000D-02

        GAM   MATRIX          3 ROWS          1 COLUMNS
 0.0
 3.0480000D-03
 2.0000000D-02

          H   MATRIX          1 ROWS          3 COLUMNS
 0.0                   0.0               1.0000000D 00
             12   ITERATIONS

      P       MATRIX          3 ROWS          3 COLUMNS
 9.1510898D-05         7.8498278D-05     1.2529794D-03
 7.8498278D-05         1.5959324D-04     9.9263243D-04
 1.2529794D-03         9.9263243D-04     2.8361755D-02

      K       MATRIX          3 ROWS          1 COLUMNS
 1.2529794D-01
 9.9263243D-02
 2.8361755D 00
             10   ITERATIONS

      P+M     MATRIX          3 ROWS          3 COLUMNS
 2.0091399D-04         4.3808351D-10    -4.5078455D-04
 4.3808351D-10         3.7590946D-04    -4.5500935D-03
-4.5078455D-04        -4.5500935D-03     4.7922265D-01
```

```
    CVU    MATRIX          1 ROWS          1 COLUMNS
 9.41591890-05

    CVHX   MATRIX          1 ROWS          1 COLUMNS
 4.79222650-01

THE INDEX OF PERFORMANCE, J:      0.086

THE ZEROS OF: DET(SI-(A-BL))

          REAL            IMAGINARY

       -0.62793D 01      0.0
       -0.73428D 00     -0.32735D 00
       -0.73428D 00      0.32735D 00

THE ZEROS OF: DET(SI-(A-KH))

          REAL            IMAGINARY

       -0.21281D 01      0.0
       -0.56406D 00     -0.14101D 01
       -0.56406D 00      0.14101D 01

THE ZEROS OF: THE DENOMINATOR POLYNOMIAL OF U(S)/Z(S)

          REAL            IMAGINARY

       -0.86889D 01      0.0
       -0.94757D 00     -0.25163D 01
       -0.94757D 00      0.25163D 01

     THE COEFFICIENTS OF THE POLYNOMIAL IN INCREASING POWERS OF S

   0.62816D 02   0.23696D 02   0.10584D 02   0.10000D 01

THE ZEROS OF: THE NUMERATOR POLYNOMIALS OF U(S)/Z(S)

I=  1      J=  1

     THE ZEROS OF THIS ELEMENT

          REAL            IMAGINARY

       -0.22480D 01     -0.39099D 01
       -0.22480D 01      0.39099D 01

     THE COEFFICIENTS OF THE POLYNOMIAL IN INCREASING POWERS OF S

   0.11528D 04   0.25481D 03   0.56675D 02
```

APPENDIX

Program Listing

OSPAC

OPTIMAL SYNTHESIS PROGRAM FOR AUTOMATIC CONTROL

THIS PROGRAM MAKES EXTENSIVE USE OF THE VARIABLE DIMENSION AUTOMATIC SYNTHESIS PROGRAM (VASP) CONFIGURED BY JOHN S. WHITE, ET. AL., AMES RESEARCH CENTER, OCTOBER 1971. A DOCUMENTATION REPORT (N72-10190) MAY BE OBTAINED FROM THE NTIS FOR VASP.

OSPAC SOLVES THE FILTER, CONTROL, AND STATE COVARIANCE MATRIX RICCATI EQUATIONS FOR A SYSTEM DEFINED BY:

$$\dot{X}=AX+BU+GAMW$$

$$Y=CX$$

$$Z=HX+V$$

IN ADDITION, THE FILTER AND CONTROL CHARACTERISTIC ROOTS ARE FOUND ALONG WITH THE TRANSFER MATRIX U(S)/Z(S)

INPUT MATRICES

A	N BY N	PLANT
B	N BY P	CONTROL
C	Q BY N	OUTPUT
F	T BY T	DISTURBANCE COVARIANCE
G	U BY U	MEASUREMENT NOISE COVARIANCE
H	U BY N	OBSERVATION
R	P BY P	CONTROL COST WEIGHTING
Q	Q BY Q	OUTPUT COST WEIGHTING
GAM	N BY T	DISTURBANCE

OUTPUT MATRICES

P	N BY N	ERROR COVARIANCE (FILTER RICCATI SOLUTION)
S	N BY N	(CONTROL RICCATI SOLUTION)
M	N BY N	(COVARIANCE INTERMEDIATE SOLUTION)
P+M	N BY N	STATE COVARIANCE
K	N BY U	OPTIMAL ESTIMATOR GAIN (FILTER)
L	P BY N	OPTIMAL CONTROLLER GAIN (CONTROL)
CVHX	U BY U	COVARIANCE OF HX
CVU	P BY P	COVARIANCE OF U

SYSTEM VECTORS

X	N BY 1	STATE
U	P BY 1	INPUT
Y	Q BY 1	OUTPUT
V	U BY 1	MEASUREMENT NOISE
W	T BY 1	SYSTEM DISTURBANCE
Z	U BY 1	MEASUREMENT

OUTPUT SCALARS

J INDEX OF PERFORMANCE

SYSTEM CHARACTERISTIC ROOTS AND ELEMENTS OF U(S)/Z(S) TRANSFER MATRIX

SCALAR SOLUTION-CONTROL PARAMETERS

NSOL NUMBER OF PROBLEMS TO BE RUN (I1)
NOPT TYPE OF PROBLEM TO BE RUN (I1)
- =1 CONTROL ONLY (L,S)
- =2 FILTER ONLY (K,P)
- =3 EVERYTHING (L,S,K,P,M,MP,Z1,J)
- =4 SAME AS 3 BUT WITH CHARACTERISTIC ROOTS AND TRANSFER MATRIX

```
C
C       VECTOR SOLUTION-CONTROL PARAMETERS
C
C         NCONT(I), I=1,2,3  (3I10)
C           NCONT(1)=1
C           NCONT(2)=THE MAXIMUM NUMBER OF STEPS
C           NCONT(3)=1
C
C
C        WITH THE EXCEPTION OF NCONT, EACH MATRIX INPUT REQUIRES A HEADER
C   CARD OF FORMAT (A4,4X,2I4) (A 4-CHARACTER TITLE, 4 BLANKS, AND THE
C   NUMBER OF ROWS AND COLUMNS IN THE MATRIX).  EACH ROW, BEGINNING ON
C   EACH ROW.  HOWEVER, EACH MATRIX ROW MUST BEGIN ON A NEW CARD.
C                       ---- ------ --- ---- ----- -- - --- ----
C   THE FORMAT FOR INPUT MATRICES IS (7E10.5)
C
C
C       MAXIMUM DIMENSIONS:
C
C             N=10            NOTE:  THE USER MUST ENSURE THAT Q,T, AND U
C             P=10                   ARE EACH LESS THAN N IN HIS PROGRAM
C             Q=9
C             T=9
C             U=9
C
C       INPUT CARD DECK ARRANGEMENT
C
C         CARD #
C              1     NSOL  (I1)
C              2     TITLE FOR PROBLEM (A72)
C              3     NOPT  (I1)
C             4+
C                    MATRICES REQUIRED (ZZ CARDS)
C                     NOPT=1  NCONT,A,B,C,Q,R
C                     NOPT=2  NCONT,A,F,G,GAM,H
C                     NOPT=3  NCONT,A,B,C,Q,R,NCONT,F,G,GAM,H,NCONT
C                     NOPT=4  SAME AS 3
C           4+ZZ      TITLE FOR NEXT PROBLEM
C              ETC.
C
C
C          ..................................................
C
C
C
C
C                              ***************
C                              *             *
C                              *  DIMENSION  *
C                              *             *
C                              ***************
C
      DOUBLE PRECISION  A(10,10),AT(10,10),B(10,10),C(10,10),D(10,10),
     1                  G(10,10),H(10,10),R(10,10),Q(10,10),GAM(10,10),
     2                  P(10,10),S(10,10),M(10,10),MP(10,10),Z1(10,10),
     3                  K(10,10),L(10,10),Z(20,20),PHI(20,20),
     4                  PHIT(20,20),F(10,10),RI(10,10),GI(10,10),
     5                  DUM(1000),CT(10,10),J,TRA,TRB,GAMT(10,10),
     6                  HT(10,10),KT(10,10),U(20),V(20),W3(10),W4(10),
     7                  ZED(10,10,10),CH(20),DC(10,10),LT(10,10),
     8                  MLT(10,10),CVU(10,10)
C
      DIMENSION  NA(2),NAT(2),NB(2),NC(2),ND(2),NG(2),NH(2),NR(2),NQ(2),
     1           NGAM(2),NP(2),NS(2),NM(2),NMP(2),NZ1(2),NK(2),NL(2),
     2           NZ(2),NPHI(2),NPHIT(2),NF(2),NRI(2),NGI(2),NCT(2),
     3           NCONT(3),NGAMT(2),NHT(2),NKT(2),NLT(2),NMLT(2),NCVU(2)
C
      CALL ERRSET (208,300,-1,1)
      COMMON  /MAX/MAXRC
      COMMON ZED
      COMMON/LINES/NLP,LIN,TITLE(23)
      KDUM = 1000
```

```
      MAXRC = 100
C   HOW MANY SOLUTIONS?
      READ (5,13) NSOL
C
      DO 12 NTIMES=1,NSOL
      CALL RDTITL
      READ (5,13) NOPT
C
C
C                          *************************
C                          *                       *
C                          *  CONTROLLER SOLUTION  *
C                          *                       *
C                          *************************
C
C
      IF (NOPT.EQ.2) GO TO 1
      READ (5,14) (NCONT(I),I=1,3)
      CALL READ (5,A,NA,B,NB,C,NC,Q,NQ,R,NR)
      NS(1) = NA(1)
      NS(2) = NA(2)
C FIND R INVERSE
      CALL EQUATE (R,NR,RI,NRI)
      CALL INV (RI,NRI,DET,DUM)
C SET S=0
      CALL SCALE (S,NA,S,NS,0.D00)
C FIND EXPONENT, FIND RI*B TRANSPOSE
      CALL AUG (A,NA,B,NB,RI,NRI,C,NC,Q,NQ,D,ND,Z,NZ,0)
C FIND EXP Z
      CALL ETPHI (Z,NZ,1.0D00,PHI,NPHI,DUM,KDUM)
C FIND RICCATI SOLUTION
      CALL RICAT (PHI,NPHI,D,ND,NCONT,L,NL,S,NS,DUM,KDUM)
      CALL PRNT (S,NS,4HS    ,1)
      CALL PRNT (L,NL,4HL    ,1)
C CHECK NOPT TO SEE IF DONE
      IF (NOPT.EQ.1) GO TO 12
C ELSE, DO FILTER SOLUTION
C    READ ADDITIONAL CONTROL AND MATRIX CARDS REQUIRED
      READ (5,14) (NCONT(I),I=1,3)
      CALL READ (4,F,NF,G,NG,GAM,NGAM,H,NH,H,NH)
      GO TO 2
C
C
C                              *********************
C                              *                   *
C                              *  FILTER SOLUTION  *
C                              *                   *
C                              *********************
C
C
    1 READ (5,14) (NCONT(I),I=1,3)
      CALL READ (5,A,NA,F,NF,G,NG,GAM,H,NH)
C FIND A TRANSPOSE
    2 CALL TRANP (A,NA,AT,NAT)
      CALL TRANP (GAM,NGAM,GAMT,NGAMT)
      CALL TRANP (H,NH,HT,NHT)
C FIND G INVERSE
      CALL EQUATE (G,NG,GI,NGI)
      CALL INV (GI,NGI,DET,DUM)
C SET P=0
      CALL SCALE (P,NA,P,NP,0.D00)
C FIND EXPONENT, G INVERSE* H
      CALL AUG (AT,NAT,HT,NHT,GI,NGI,GAMT,NGAMT,F,NF,D,ND,Z,NZ,0)
C FIND EXP Z
      CALL ETPHI (Z,NZ,1.D00,PHI,NPHI,DUM,KDUM)
C FIND RICCATI SOLUTION
      CALL RICAT (PHI,NPHI,D,ND,NCONT,KT,NKT,P,NP,DUM,KDUM)
      CALL TRANP (KT,NKT,K,NK)
      CALL PRNT (P,NP,4HP    ,1)
      CALL PRNT (K,NK,4HK    ,1)
C CHECK NOPT TO SEE IF DONE
      IF (NOPT.EQ.2) GO TO 12
C
C
```

```
C                                ********************************
C                                *                              *
C                                *  STATE COVARIANCE SOLUTION   *
C                                *                              *
C                                ********************************
C
C
      READ (5,14) (NCONT(I),I=1,3)
C USE MATRIX F FOR   (A-BL), PHI FOR ITS TRANSPOSE
      CALL MULT (B,NB,L,NL,PHI,NPHI)
      CALL SUBT (A,NA,PHI,NPHI,F,NF)
      CALL TRANP (F,NF,PHI,NPHI)
C SETUP R AS A ZERO MATRIX
      CALL SCALE (R,NR,R,NR,0.D00)
C SET M=0
      CALL SCALE (M,NA,M,NM,0.D00)
C FIND EXPONENT
      CALL AUG (PHI,NPHI,B,NB,R,NR,KT,NKT,G,NG,D,ND,Z,NZ,0)
C FIND EXP
      CALL ETPHI (Z,NZ,1.D00,PHIT,NPHIT,DUM,KDUM)
C FIND RICCATI SOLUTION
      CALL RICAT (PHIT,NPHIT,D,ND,NCONT,Z1,NZ1,M,NM,DUM,KDUM)
C FIND P+M, PRINT
      CALL ADD (M,NM,P,NP,MP,NMP)
      CALL PRNT (MP,NMP,4HP+M ,1)
C FIND CVU, PRINT
      CALL TRANP (L,NL,LT,NLT)
      CALL MULT (M,NM,LT,NLT,MLT,NMLT)
      CALL MULT (L,NL,MLT,NMLT,CVU,NCVU)
      CALL PRNT (CVU,NCVU,4HCVU ,1)
C FIND CVHX, PRINT
      CALL MULT (H,NH,MP,NMP,PHI,NPHI)
      CALL MULT (PHI,NPHI,HT,NHT,PHIT,NPHIT)
      CALL PRNT (PHIT,NPHIT,4HCVHX,1)
C     FIND J
      CALL TRANP (C,NC,CT,NCT)
      CALL MULT (CT,NCT,Q,NQ,PHI,NPHI)
      CALL MULT (PHI,NPHI,C,NC,PHIT,NPHIT)
      CALL MULT (PHIT,NPHIT,P,NP,Z,NZ)
      CALL MULT (S,NS,P,NP,PHI,NPHI)
      CALL TRANP (H,NH,HT,NHT)
      CALL MULT (PHI,NPHI,HT,NHT,PHIT,NPHIT)
      CALL MULT (PHIT,NPHIT,GI,NGI,PHI,NPHI)
      CALL MULT (PHI,NPHI,H,NH,PHIT,NPHIT)
      CALL MULT (PHIT,NPHIT,P,NP,PHI,NPHI)
      CALL TRCE (Z,NZ,TRA)
      CALL TRCE (PHI,NPHI,TRB)
      J = TRA+TRB
      CALL LNCNT (2)
      WRITE (6,15) J
      IF (NOPT.LT.4) GO TO 12
C
C
C                     ************************************
C                     *                                  *
C                     *  OBTAIN ROOTS OF DET(SI-(A-BL))  *
C                     *                                  *
C                     ************************************
C
C
C     SINCE MATRIX F ALREADY EQUALS (A-BL) PROCEED DIRECTLY
      CALL DIMCH (F,NA,DC)
      CALL CHREQA (DC,NA(1),CH)
      CALL PROOT (NA(1),CH,U,V,1)
      WRITE (6,16)
      WRITE (6,17)
      WRITE (6,18)
      II = NA(1)
C
      DO 3 I=1,II
      WRITE (6,19) U(I),V(I)
    3 CONTINUE
C
```

```
C
C             **********************************
C             *                                *
C             *  OBTAIN ROOTS OF DET(SI-(A-KH))  *
C             *                                *
C             **********************************
C
C
C     FIND K*H, PUT IN G
      CALL MULT (K,NK,H,NH,G,NG)
C     FIND A-KH, PUT IN C
      CALL SUBT (A,NA,G,NG,C,NC)
C     FIND ROOTS AND PRINT
      CALL DIMCH (C,NC,DC)
      CALL CHREQA (DC,NC(1),CH)
      CALL PROOT (NC(1),CH,U,V,1)
      WRITE (6,16)
      WRITE (6,20)
      WRITE (6,18)
      JJ = NC(1)
C
      DO 4 I=1,JJ
      WRITE (6,19) U(I),V(I)
    4 CONTINUE
C
C            ************************************
C            *                                  *
C            *  OBTAIN U(S)/Z(S) TRANSFER MATRIX  *
C            *                                  *
C            ************************************
C
C
C     FIND A-BL-KH, PUT IN C
      CALL SUBT (F,NA,G,NG,C,NC)
      CALL DIMCH (C,NC,DC)
      CALL CHREQ (DC,NC(1),CH,1)
      CALL PROOT (NC(1),CH,U,V,1)
      KK = NC(1)
      K2 = KK+1
      WRITE (6,16)
      WRITE (6,21)
      WRITE (6,18)
C
      DO 5 I=1,KK
    5 WRITE (6,19) U(I),V(I)
C
      WRITE (6,22)
      WRITE (6,23) (CH(I),I=1,K2)
      LL = NL(1)
C
      DO 11 K1=1,LL
      CALL DIMCH (L,NL,DC)
      MM = NL(2)
C
      DO 6 I=1,MM
    6 W3(I) = DC(I,K1)
C
      WRITE (6,16)
      WRITE (6,24)
      NN = NK(2)
C
      DO 10 J1=1,NN
      CALL DIMCH (K,NK,DC)
      IJ = NK(1)
C
      DO 7 II=1,IJ
    7 W4(II) = DC(II,J1)
C
      CALL MPY (W4,W3,NK(1),CH,NO)
      WRITE (6,25) K1,J1
      N2 = NO+1
      CALL PROOT (NO,CH,U,V,1)
```

```
      WRITE (6,26)
      WRITE (6,18)
C
      DO 8 I8=1,NO
      WRITE (6,19) U(I8),V(I8)
    8 CONTINUE
C
C
      DO 9 I=1,N2
    9 CH(I) = -CH(I)
C
      WRITE (6,22)
      WRITE (6,23) (CH(I),I=1,N2)
   10 CONTINUE
C
   11 CONTINUE
C
   12 CONTINUE
C
C
   13 FORMAT (I1)
   14 FORMAT (3I10)
   15 FORMAT ('OTHE INDEX OF PERFORMANCE, J:',3X,F 7.3)
   16 FORMAT ('OTHE ZEROS OF:')
   17 FORMAT ('+                 DET(SI-(A-BL))')
   18 FORMAT ('0',T11,'REAL',T23,'IMAGINARY',/)
   19 FORMAT (' ',T7,D14.5,T22,D14.5)
   20 FORMAT ('+                 DET(SI-(A-KH))')
   21 FORMAT ('+                 THE DENOMINATOR POLYNOMIAL OF U(S)/Z(S)')
   22 FORMAT ('0      THE COEFFICIENTS OF THE POLYNOMIAL IN INCREASING PO
     1WERS OF S')
   23 FORMAT ('0',(8D14.5))
   24 FORMAT ('+                 THE NUMERATOR POLYNOMIALS OF U(S)/Z(S)')
   25 FORMAT ('OI= ',I2,5X,'J= ',I2)
   26 FORMAT ('0      THE ZEROS OF THIS ELEMENT')
      END
```

```
      SUBROUTINE RDTITL
      COMMON /LINES/NLP,LIN,TITLE(23)
      READ  (5,100)  (TITLE(I),I=1,18)
  100 FORMAT  (18A4)
      CALL LNCNT(100)
      RETURN
      END
```

```
      SUBROUTINE  PRNT(AR,NAR,NAM,IP)
C   SUBR PRNT   PRINTS DOUBLE PRECISION MATRIX
      COMMON  /FORM/NEPR,FMT1(6),FMT2(6)
      COMMON/LINES/NLP,LIN,TITLE(23)
      COMMON  /MAX/MAXRC
C- NOTE NLP NO. LINES/PAGE VARIES WITH THE INSTALLATION.
      DATA KZ,KW,KB /1H0,1H1,1H /
      REAL*8 AR
      DIMENSION AR(1),NAR(2)
      NAME = NAM
C-IF IP =1,HEADLINE SAME PAGE, IF IP =2,  HEADLINE, NEW PAGE
C     IP=3,  NO HEADLINE, SAME PAGE,     IP=4, NO HEADLINE, NEW PAGE
      II = IP
      NR=NAR(1)
      NC=NAR(2)
      NLST = NR * NC
      IF(NLST .GT. MAXRC .OR. NLST .LT. 1.OR.NR.LT.1)  GO TO 16
      IF(NAME  .EQ. 0) NAME = KB
C- SKIP HEADLINE IF REQUESTED.
      GO TO (11,10,132,12),        II
   10 CALL LNCNT(100)
   11 CALL LNCNT(2)
    3 WRITE(6,177) KZ,NAME,NR,NC
177   FORMAT(A1,5X,A4,8H  MATRIX,5X,I3,5H ROWS,5X,I3,8H COLUMNS)
      GO TO 13
   12 CALL LNCNT(100)
      GO TO 13
  132 CALL LNCNT(2)
      WRITE (6,891)
  891 FORMAT (1H0)
C- BELOW COMPUTE NR OF LINES/ ROW --DECIDE IF 1 EXTRA BLANK LINE
   13 J=(NC-1)/NEPR+1
C- WHY ALWAYS ADD 1 LINE- BECAUSE IF MULTIPLE, USE 1 BLK LINE EXTRA.
      NLPW = J
      JST = 1
C- COMPUTE LAST ROW POSITION -1 BELOW
      NLST = NLST -NR
      MN=NC
      IF  (NC.GT.NEPR)   MN=NEPR
      KLST=NR*(MN-1)
91    CONTINUE
      DO 912 J = JST, NR
      CALL LNCNT(NLPW)
      KLST = KLST +1
      WRITE (6,FMT1)    (AR(N), N = J, KLST, NR)
      IF  (NC.LE.NEPR)   GO TO 912
      NLST = NLST +1
      KNR=KLST+NR
      WRITE (6,FMT2)(AR(N),N=KNR,NLST,NR)
912   CONTINUE
      RETURN
   16 CALL LNCNT(1)
      WRITE  (6,916)   NAM,NAR
  916 FORMAT  (' ERROR IN PRNT  MATRIX ',A4,' HAS NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE LNCNT   (N)
      COMMON/LINES/NLP,LIN,TITLE(23)
      LIN=LIN+N
      IF  (LIN.LE.NLP)    GO TO 20
      WRITE  (6,1010)  (TITLE(I),I=1,23)
 1010 FORMAT  (1H1,23A4/)
      LIN=2+N
      IF  (N.GT.NLP)  LIN=2
   20 RETURN
      END
```

```
      SUBROUTINE ADD (A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      COMMON  /MAX/MAXRC
      DOUBLE PRECISION A,B,C
      IF((NA(1).NE.NB(1)).OR.(NA(2).NE.NB(2))) GO TO 999
      NC(1)=NA(1)
      NC(2)=NA(2)
      L=NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 C(I)=A(I)+B(I)
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6,50) NA,NB
   50 FORMAT  (' DIMENSION ERROR IN ADD     NA=',2I6,5X,'NB=',2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE MULT(A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      DOUBLE PRECISION A,B,C
      COMMON  /MAX/MAXRC
      NC(1) = NA(1)
      NC(2) = NB(2)
      IF(NA(2).NE.NB(1)) GO TO 999
      NAR = NA(1)
      NAC = NA(2)
      NBC = NB(2)
      NAA=NAR*NAC
      NBB=NAR*NBC
      IF  (NAR.LT.1.OR.NAA.LT.1.OR.NAA.GT.MAXRC.OR.NBB.LT.1.OR.
     1    NBB.GT.MAXRC)  GO TO 999
      IR = 0
      IK=-NAC
      DO 300 K=1,NBC
      IK = IK + NAC
      DO 300 J=1,NAR
      IR=IR+1
      IB=IK
      JI=J-NAR
      C(IR)=0.
      DO 300 I=1,NAC
      JI = JI + NAR
      IB=IB+1
      C(IR)=C(IR)+A(JI)*B(IB)
  300 CONTINUE
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6,500) (NA(I),I=1,2),(NB(I),I=1,2)
  500 FORMAT  (' DIMENSION ERROR IN MULT    NA=',2I6,5X,'NB=',2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE SCALE (A, NA, B, NB, S)
      DIMENSION A(1),B(1),NA(2),NB(2)
      COMMON  /MAX/MAXRC
      DOUBLE PRECISION A, B, S
      NB(1) = NA(1)
      NB(2) =NA(2)
      L = NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 B(I)=A(I)*S
 1000 RETURN
  999 CALL LNCNT(1)
      WRITE  (6,50) NA
   50 FORMAT  (' DIMENSION ERROR IN SCALE    NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE TRANP(A,NA,B,NB)
      DIMENSION A(1),B(1),NA(2),NB(2)
      DOUBLE PRECISION A,B
      COMMON  /MAX/MAXRC
      NB(1)=NA(2)
      NB(2)=NA(1)
      NR=NA(1)
      NC=NA(2)
      L=NR*NC
      IF  (NR   .LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      IR=0
      DO 300 I=1,NR
      IJ=I-NR
      DO 300 J=1,NC
      IJ=IJ+NR
      IR=IR+1
  300 B(IR)=A(IJ)
      RETURN
  999 CALL LNCNT(1)
      WRITE  (6,50)  NA
   50 FORMAT  (' DIMENSION ERROR IN TRANP   NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE INV(A,NA,DET,L)
      DIMENSION A(1), L(1),NA(2)
      DOUBLE PRECISION A, DET, BIGA ,  HOLD
      COMMON  /MAX/MAXRC
      IF  (NA(1).NE.NA(2))  GO TO 600
C     SEARCH FOR LARGEST ELEMENT
      DET= 1.
      N=NA(1)
      NSQ=N*N
      IF  (N.LT.1.OR.NSQ.GT.MAXRC)   GO TO 600
      NK = - N
      DO 80 K= 1, N
      NK = NK + N
      L(K) = K
      NPK=N+K
      L(NPK)=K
      KK = NK + K
      BIGA = A(KK)
      DO 20 J= K, N
      IZ = N*(J - 1)
      DO   20 I= K, N
      IJ = IZ + I
   10 IF(DABS(BIGA) - DABS(A(IJ))) 15, 20, 20
   15 BIGA = A(IJ)
      L(K) = I
      NPK=N+K
      L(NPK)=J
   20 CONTINUE
C     INTERCHANGE ROWS
      J = L(K)
      IF(J - K) 35, 35, 25
   25 KI = K - N
      DO 30 I = 1, N
      KI = KI + N
      HOLD = -A(KI)
      JI = KI - K + J
      A(KI) = A(JI)
   30 A(JI) = HOLD
C     INTERCHANGE COLUMNS
   35 NPK=N+K
      I=L(NPK)
      IF (I - K) 45, 45, 38
   38 JP = N*(I - 1)
      DO 40 J= 1, N
      JK = NK + J
      JI = JP + J
      HOLD = -A(JK)
      A(JK) = A(JI)
   40 A(JI) = HOLD
C     DIVIDE COLUMN BY MINUS PIVOT(VALUE OF PIVOT ELEMENTS IS CONTAINE
C     IN BIGA)
   45 IF(BIGA) 48, 46, 48
   46 DET = 0.
      CALL LNCNT (1)
      KKK=KK-1
      WRITE  (6,1046)  KKK
 1046 FORMAT  (' INV ERROR  DETERMINANT OF A=0   RANK OF A=',I4)
      CALL ASPERR
      RETURN
   48 DO 55 I= 1, N
      IF (I - K) 50, 55, 50
   50 IK = NK + I
      A(IK) =-A(IK)/(BIGA)
   55 CONTINUE
C     REDUCE MATRIX
      DO 65 I= 1, N
      IK = NK + I
      HOLD = A(IK)
      IJ = I - N
      DO 65 J= 1, N
      IJ = IJ + N
      IF(I  - K ) 60, 65, 60
   60 IF(J- K) 62, 65, 62
```

```
   62 KJ = IJ -I + K
      A(IJ) = HOLD* A(KJ) + A(IJ)
   65 CONTINUE
C     DIVIDE ROW BY PIVOT
      KJ = K - N
      DO 75 J= 1, N
      KJ = KJ + N
      IF(J - K) 70, 75, 70
   70 A(KJ) = A(KJ)/BIGA
   75 CONTINUE
C     PRODUCT OF PIVOTS
      DET=DET*BIGA
C     REPLACE PIVOT BY RECIPROCAL
      A(KK) = 1./BIGA
   80 CONTINUE
C     FINAL ROW AND COLUMN INTERCHANGE
      K = N
  100 K = K - 1
      IF(K) 150, 150, 105
  105 I = L(K)
      IF (I - K ) 120, 120, 108
  108 JQ = N*( K - 1)
      JR = N*(I- 1)
      DO 110 J= 1, N
      JK = JQ + J
      HOLD = A(JK)
      JI = JR + J
      A(JK) = - A(JI)
  110 A(JI) = HOLD
  120 NPK=N+K
      J=L(NPK)
      IF(J - K) 100, 100, 125
  125 KI = K - N
      DO 130 I= 1, N
      KI = KI + N
      HOLD = A(KI)
      JI = KI - K + J
      A(KI) = - A(JI)
  130 A(JI) = HOLD
      GO TO 100
  150 RETURN
  600 CALL LNCNT (1)
      WRITE  (6,1600)  NA
 1600 FORMAT  (' INV REQUIRES SQUARE MATRIX   NA=',2I4)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE NORM(A,NA,ANORM)
      DIMENSION NA(2),A(1)
      DOUBLE PRECISION A,ANORM,SUM,ROWMAX,COLMAX
      COMMON  /MAX/MAXRC
      NAR = NA(1)
      NAC = NA(2)
      L=NAR*NAC
      IF  (NAR  .LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      COLMAX = 0.
      ROWMAX = 0.
      K = 0
      DO 300 I = 1,NAC
      SUM = 0.
      DO 301 J = 1,NAR
      K = K + 1
  301 SUM = SUM + DABS(A(K))
      IF (COLMAX.LT.SUM) COLMAX = SUM
  300 CONTINUE
      DO 302 I = 1,NAR
      SUM = 0.
      K = I - NAR
      DO 303 J = 1,NAC
      K = K + NAR
  303 SUM = SUM + DABS(A(K))
      IF (ROWMAX.LT.SUM) ROWMAX=SUM
  302 CONTINUE
      ANORM = DMIN1(COLMAX,ROWMAX)
      RETURN
  999 CALL LNCNT (1)
      WRITE  (6,50)  NA
   50 FORMAT  (' DIMENSION ERROR IN NORM     NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE UNITY(A,NA)
      DIMENSION A(1),NA(2)
      DOUBLE PRECISION A
      IF(NA(1).NE.NA(2)) GO TO 999
      CALL SCALE(A,NA,A,NA,0.DO)
      J = - NA(1)
      NAX = NA(1)
      DO 300 I=1,NAX
      J=NAX +J+1
  300 A(J)=1.
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6, 50)(NA(I),I=1,2)
   50 FORMAT  (' DIMENSION ERROR IN UNITY    NA=',2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE EQUATE(A,NA,B,NB)
      DIMENSION A(1),B(1),NA(2),NB(2)
      DOUBLE PRECISION A, B
      COMMON  /MAX/MAXRC
      NB(1) = NA(1)
      NB(2) =NA(2)
      L=NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 B(I)=A(I)
 1000 RETURN
  999 CALL LNCNT (1)
      WRITE  (6,50)  NA
   50 FORMAT  (' DIMENSION ERROR IN EQUATE  NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE ETPHI(A,NA,TT,B,NB,DUMMY,KDUM)
      DIMENSION A(1),B(1),DUMMY(1),NA(2),NB(2),ND(2)
      DOUBLE PRECISION A, T, TT, ANAA, TMAX, B, DUMMY ,S, SC
      COMMON  /MAX/MAXRC
      NR=NA(1)
      NCC=NA(2)
      NB(1)=NR
      NB(2)=NCC
      LD=NR*NCC
      IF (NR.NE.NCC.OR.NR.LT.1            .OR.LD.GT.MAXRC) GO TO 998
      NDD=2*NA(1)*NA(1)
      IF(KDUM  .LT.NDD) GO TO 998
      NDD= NA(1)*NA(1)+1
      T=TT
      CALL NORM(A,NA,ANAA)
      TMAX= 10.01/ANAA
      K=0
  101 IF (TMAX-T ) 103,104,104
  103 K=K+1
      T=TT/2**K
      IF (K-1000) 101,102,102
  104 SC=T
      CALL SCALE(A,NA,A,NA,T)
      CALL UNITY(B,NB)
      II=2
      N = 35
      CALL ADD(A,NA,B,NB,DUMMY(1),ND)
      CALL EQUATE(A,NA,DUMMY(NDD),ND)
  106 CALL MULT(A,NA,DUMMY(NDD),ND,B,NB)
      S=1.D0/II
      CALL SCALE(B,NB,DUMMY(NDD),ND,S)
      CALL ADD(DUMMY(NDD),ND,DUMMY(1),ND,B,NB)
      CALL EQUATE(B,NB,DUMMY(1),ND)
      N=N-1
      IF (N) 107,107,105
  105 II=II+1
      GO TO 106
  107 IF (K) 109,108,212
  109 CALL LNCNT (1)
      WRITE (6,110)
  110 FORMAT (' ERROR IN ETPHI  K IS NEGATIVE')
  112 IF (K-1) 213,212,212
  213 K=1
  212 DO 111 J=1,K
      T=2*T
      CALL EQUATE(B, NB, DUMMY(1), ND)
      CALL EQUATE(DUMMY(1), ND, DUMMY(NDD), ND)
  111 CALL MULT(DUMMY(NDD),ND,DUMMY(1),ND,B,NB)
      TT=T
  108 CONTINUE
      S=1.D0/SC
      CALL SCALE(A,NA,A,NA,S)
      RETURN
  102 CALL LNCNT (1)
      WRITE (6,119)
  119 FORMAT   (' ERROR IN ETPHI   K=1000')
      CALL ASPERR
      RETURN
  998 CALL LNCNT (1)
      WRITE  (6,50)  NA,KDUM,NDD
   50 FORMAT  (' DIMENSION ERROR IN ETPHI   NA=',2I6,   'KDUM=',I5,5X,
     1   'KDUM(MIN)=',I5)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE AUG(F,NF,G,NG,RI,NRI,H, NH,Q,NQ, C,NC,Z,NZ, II)
      DIMENSION F(1),G(1),RI(1),H(1),Q(1),Z(1),C(1)
      DIMENSION NNP1(2),NNP2(2),NNP3(2),NNP4(2),NF(2),NG(2),NRI(2),
     1NH(2),NZ(2),NC(2),NN(2),NQ(2)
      DOUBLE PRECISION F, G, RI,H,Q,C,Z
      IF((NF(1).NE.NF(2)).OR.(NRI(1).NE.NRI(2)).OR.(
     1NQ(1).NE.NQ(2))) GO TO 995
      NNZ=2*NF(1)
      IF( (NG(1).NE.NF(1)).OR.(NG(2).NE.NRI(1)))GO TO 995
      IF(II.EQ.1) GO TO 206
      IF((NH(1).NE.NQ(1)).OR.(NH(2).NE.NF(2))) GO TO 995
  206 TWO = 2
      N = NF(1)
      NSQ = N*N
      NZ(1)=NNZ
      NZ(2)=NNZ
      NP1=1
      NP2 = NP1 + NSQ
      NP3 = NP2+NSQ
      NP4 = NP3 + NSQ
      CALL TRANP(G,NG,Z(NP4),NNP4)
      CALL MULT(RI,NRI,Z(NP4),NNP4,C,NC)
      CALL MULT(G,NG,C,NC,Z(NP4), NNP4)
      IF(II .EQ. 1) GO TO 204
      CALL TRANP(H,NH,Z(NP3), NNP3)
      CALL MULT(Q,NQ,H,NH,Z(1), NNP1)
      CALL MULT(Z(NP3),NNP3,Z(   1),NNP1,Z(NP2),NNP2)
      GO TO 205
  204 CALL EQUATE(Q, NQ, Z(NP2), NQ)
  205 NPAIR=MOD(N,2)
      IF(NPAIR.EQ.0) NPAIR=TWO
      NN(1) = N
      NN(2) = 1
      GO TO (201,202),NPAIR
  201 DO 300 I=1,N,2
      NP2 = N*(N+I-1)+1
      NTH3=TWO*N*(I-1)+N+1
  300 CALL EQUATE(Z(NP2),NN,Z(NTH3),NN)
      DO 302 I=2,N,2
      NP4=N*(3*N+I-1)+1
      NTH2=TWO*N*(N+I-1)+1
  302 CALL EQUATE(Z(NP4),NN,Z(NTH2),NN)
      GO TO (202,203),NPAIR
  202 DO 301 I=2  ,N,2
      NP2 = N*(N+I-1)+1
      NTH3=TWO*N*(I-1)+N+1
  301 CALL EQUATE(Z(NP2),NN,Z(NTH3),NN)
      DO 304 I=1,N,2
      NP4=N*(3*N+I-1)+1
      NTH2=TWO*N*(N+I-1)+1
  304 CALL EQUATE(Z(NP4),NN,Z(NTH2),NN)
      GO TO (203,201),NPAIR
  203 DO 303 I=1,N
      IJ = I+N
      DO 303 J=1,N
      JJ = J+N
      L1=NNZ*(J-1)+I
      L2=NNZ*(IJ-1)+JJ
      L3=N*(J-1)+I
      Z(L1)=-F(L3)
  303 Z(L2)=F(L3)
      GO TO 1000
  995 CALL LNCNT (2)
      WRITE  (6,50)  NF,NG,NRI,NH,NQ
   50 FORMAT  (' DIMENSION ERROR IN AUG',T35,'NF',9X,'NG',9X,'NRI',8X,
     1   'NH',9X,'NQ'/29X,5(3X,2I6))
  999 CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE RICAT(PHI,NPHI,C,NC,NCONT,K,NK,PT,NPT,W,KDUM)
      DIMENSION NCONT(3),NPHI(2),NC(2),NK(2),NN(2),NM(2),NPT(2)
      DIMENSION PHI(1),C(1),K(1),PT(1),W(1)
      DOUBLE PRECISION PHI, C, K, PT, SUM, SUMN, AL, W,DET
      TWO=2
      N = NPHI(1)/TWO
      NSQ=N*N
      NSQ4=4*NSQ
      NP1=1
      NP2= NSQ+NP1
      NP3=NSQ+NP2
      NP4= NSQ+NP3
      IF  (KDUM.LT.NSQ4 )  GO TO 600
      IF  (NPHI(2).NE.NPHI(1).OR.NC(2).NE.N.OR.NPT(1).NE.N.OR.NPT(2).
     1  NE.N)   GO TO 600
      NPRINT=NCONT(1)
      NBLOCK=NCONT(2)/NPRINT
      NZ=NCONT(3)
C     REARRANGE PHI MATRIX
      NN(1)=N
      NN(2)=1
      DO 300 I=1,N
      NTH1 =TWO*N*(I-1)+1
      NTH3=NTH1+N
      NW1=N*(I-1)+1
      NW2=NW1+N*N
      CALL  EQUATE(PHI(NTH1),NN,W(NW1),NN)
  300 CALL  EQUATE(PHI(NTH3),NN,W(NW2),NN)
      NM(1)=TWO*N*N
      NM(2)=1
      CALL  EQUATE(W(1),NM,PHI(1),NM)
      DO 301 I=1,N
      NTH2=TWO*N*(N+I-1)+1
      NTH4=NTH2+N
      NW3 = N*(TWO*N+I-1)+1
      NW4= NW3+N*N
      CALL  EQUATE(PHI(NTH2),NN,W(NW3),NN)
  301 CALL  EQUATE(PHI(NTH4),NN,W(NW4),NN)
      NWW=TWO*N*N+1
      CALL  EQUATE(W(NWW),NM,PHI(NWW),NM)
C     MAIN COMPUTATION
C     CALL UNITY(PT,NPT)
      DO 306 I= 1,N
  306 K(I) = 0.
      NTOT=0
      DO 403 I=1,NBLOCK
      DO 104 J=1,NPRINT
      CALL  MULT(PHI(NP3), NPT, PT, NPT, W(1), NPT)
      CALL  ADD (PHI(1), NPT, W(1), NPT, W(1), NPT)
      CALL  INV(W(1), NPT, DET, W(NP2))
      CALL  MULT(PHI(NP4), NPT, PT, NPT, W(NP2), NPT)
      CALL  ADD(PHI(NP2), NPT, W(NP2), NPT, W(NP2), NPT)
      CALL  MULT(W(NP2), NPT, W(1), NPT, PT, NPT)
      SUMN=0.
      SUM=0.
      DO 202 IL=1,N
      ILP=IL+NP3
      NIL=(IL-1)*N+IL
      SUM=SUM+DABS(PT(NIL))
  202 SUMN=SUMN+DABS(PT(NIL)          -W(ILP))
      AL=SUMN/SUM
      DO 201 IL=1,N
      NIL=(IL-1)*N+IL
      ILP=IL+NP3
  201 W(ILP)    =PT(NIL)
  204 DO 104 M=2,N
      N1=M-1
      DO 104 L=1,N1
      L1=N*(L-1)+M
      L2=N*(M-1)+L
      PT(L1)=(PT(L1) + PT(L2))/2.
      PT(L2)=PT(L1)
      IF(AL-.00001) 203,203,104
```

```
 104 CONTINUE
     NTOT=I*NPRINT
     GO TO 305
 203 NTOT=NTOT+J
 305 CALL  MULT (C,NC,PT,NPT,K,NK)
 103 GO TO (404,400,401,402), NZ
 400 CALL LNCNT (1)
     WRITE (6, 50) NTOT
  50 FORMAT (10X,I4,14H   ITERATIONS  )
     CALL PRNT  (PT,NPT,'P(T)',1)
     GO TO 403
 401 CALL LNCNT (1)
     WRITE (6, 50) NTOT
     CALL PRNT   (K,NK,'K(T)',1)
     GO TO 403
 402 CALL LNCNT (1)
     WRITE (6, 50) NTOT
     CALL PRNT   (K,NK,'K(T)',1)
     CALL PRNT  (PT,NPT,'P(T)',1)
     IF(AL-.00001) 210,210,403
 404 IF(AL-.00001) 405,405,403
 403 CONTINUE
 405 CALL LNCNT (1)
     WRITE(6,50)NTOT
C    REARRANGE PHI MATRIX
 210 CALL EQUATE(PHI(1),NM,W(1),NM)
     DO 303 I=1,N
     NTH1 =TWO*N*(I-1)+1
     NTH3=NTH1+N
     NW1=N*(I-1)+1
     NW2=NW1+N*N
     CALL EQUATE(W(NW1),NN,PHI(NTH1),NN)
 303 CALL EQUATE(W(NW2),NN,PHI(NTH3),NN)
     CALL EQUATE(PHI(NWW),NM,W(NWW),NM)
     DO 304 I=1,N
     NTH2=TWO*N*(N+I-1)+1
     NTH4=NTH2+N
     NW3 = N*(TWO*N+I-1)+1
     NW4= NW3+N*N
     CALL EQUATE(W(NW3),NN,PHI(NTH2),NN)
 304 CALL EQUATE(W(NW4),NN,PHI(NTH4),NN)
     RETURN
 600 CALL LNCNT (2)
     WRITE  (6,1600)  NPHI,NC,NPT,KDUM,NS04
1600 FORMAT  (' DIMENSION ERROR IN RICAT',T35,'NPHI',7X,'NC',9X,'NPT'
   1    ,6X,'KDUM',3X,'KDUM(MIN)'/29X,3(3X,2I4),4X,I4,5X,I4)
     CALL ASPERR
     RETURN
     END
```

```
      SUBROUTINE ASPERR
      DATA I /10/
      CALL TRACE
C    ERRTRA IS THE 360/67 TRACE ROUTINE      TRACE IS FOR TSS
C     CALL ERRTRA
C     THIS IS AN INSTALLATION DEPENDENT SUBROUTINE
C     SUBROUTINE ERRTRA IS A SUBROUTINE SUPPLIED BY THE AMES OPERATING
C          SYSTEM TO PROVIDE AN ERROR WALKBACK
C     THE STATEMENT  "CALL ERRTRA" SHOULD BE EITHER
C          1) CHANGED TO MATCH THE USERS OPERATING SYSTEM,
C       OR 2) DELETED ALTOGETHER.
      I=I-1
      IF (I.GT.0)  RETURN
      I=10
      WRITE (6,100)
  100 FORMAT (' TOO MANY ERRORS.   EXIT CALLED')
      CALL EXIT
      RETURN
      END
```

```
      BLOCK DATA
      COMMON  /FORM/NEPR,FMT1(6),FMT2(6)
      COMMON/LINES/NLP,LIN,TITLE(23)
      COMMON  /MAX/MAXRC
      DATA  MAXRC/6400/
C- NOTE NLP NO. LINES/PAGE VARIES WITH THE INSTALLATION.
      DATA LIN,NLP/1,74/
      DATA  NEPR,FMT1     /7,'(1P7','D16.','7)'/
      DATA  FMT2/'(3X,','1P7D','16.7',')'/
      DATA  TITLE  /19*'    ','OSPA','C PR','OGRA','M   '/
      END
```

```
      SUBROUTINE READ1 (A,NA,NZ,NAM)
      COMMON  /MAX/MAXRC
      DIMENSION  A(1  ),NA(2),NZ(2)
      DOUBLE PRECISION A
      IF  (NZ(1).EQ.0)  GO TO 410
      NR=NZ(1)
      NC=NZ(2)
      NLST=NR*NC
      IF(NLST .GT. MAXRC .OR. NLST .LT. 1.OR.NR.LT.1)  GO TO 16
      DO 400 I = 1, NR
  400 READ (5,101) (A(  J), J = I,NLST,NR)
      NA(1)=NR
      NA(2)=NC
  410 CALL  PRNT (A,NA,NAM,1)
  101 FORMAT (7F10.5)
      RETURN
   16 CALL LNCNT(1)
      WRITE  (6,916)  NAM,NR,NC
  916 FORMAT  (' ERROR IN READ1   MATRIX ',A4,' HAS NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINE READ (I, A, NA, B, NB, C, NCX, D, ND, E, NE)
      DIMENSION A(1), B(1), C(1), D(1), E(1)
      DIMENSION NA(2), NB(2),NCX(2), ND(2), NE(2),NZ(2)
      DOUBLE PRECISION A, B, C, D, E
      READ(5,100) LAB,                NZ(1), NZ(2)
      CALL READ1(A, NA,NZ,  LAB)
      IF(I .EQ. 1) GO TO 999
      READ(5,100) LAB,                NZ(1), NZ(2)
      CALL READ1(B, NB,NZ,  LAB)
      IF(I .EQ. 2) GO TO 999
      READ(5,100) LAB,                NZ(1), NZ(2)
      CALL READ1(C, NCX,NZ, LAB)
      IF(I .EQ. 3) GO TO 999
      READ(5,100) LAB,                NZ(1), NZ(2)
      CALL READ1(D, ND,NZ,  LAB)
      IF(I .EQ. 4) GO TO 999
      READ(5,100) LAB,                NZ(1), NZ(2)
      CALL READ1(E, NE,NZ,  LAB)
  100 FORMAT(A4,4X,2I4)
  999 RETURN
      END
```

```
      SUBROUTINE SUBT(A,NA,B,NB,C,NC)
      DIMENSION A(1),B(1),C(1),NA(2),NB(2),NC(2)
      DOUBLE PRECISION A,B,C
      COMMON  /MAX/MAXRC
      IF((NA(1).NE.NB(1)).OR.(NA(2).NE.NB(2))) GO TO 999
      NC(1)=NA(1)
      NC(2)=NA(2)
      L=NA(1)*NA(2)
      IF  (NA(1).LT.1.OR.L.LT.1.OR.L.GT.MAXRC)  GO TO 999
      DO 300 I=1,L
  300 C(I)=A(I)-B(I)
      GO TO 1000
  999 CALL LNCNT (1)
      WRITE(6,50) NA,NB
   50 FORMAT  (' DIMENSION ERROR IN SUBT    NA=',2I6,5X,'NB=',2I6)
      CALL ASPERR
 1000 RETURN
      END
```

```
      SUBROUTINE TRCE  (A,NA,TR)
      DOUBLE PRECISION A(1),TR
      DIMENSION NA(2)
      COMMON  /MAX/MAXRC
      IF (NA(1).NE.NA(2)) GO TO 600
      TR=0.D0
      N=NA(1)
      NN=N*N
      IF  (N.LT.1.OR.NN.GT.MAXRC)   GO TO 600
      DO 10 I=1,N
      M=I+N*(I-1)
   10 TR=TR+A(M)
      RETURN
  600 CALL LNCNT(1)
      WRITE (6,1600) NA
 1600 FORMAT (' TRACE REQUIRES SQUARE MATRIX    NA=',2I6)
      CALL ASPERR
      RETURN
      END
```

```
      SUBROUTINEPROOT(N,A,U,V,IR)
C   THIS SUBROUTINE USES A MODIFIED BARSTOW METHOD TO FIND THE
C     ROOTS OF A POLYNOMIAL.
      DOUBLE PRECISION A(20),U(20),V(20),H(21),B(21),C(21),P,Q,R,F,E,
     1                 CBAR,D,QP,PP,ZZ
      DO 91 I=1,N
      U(I)=0.
      V(I)=0.
   91 CONTINUE
      IREV=IR
      NC=N+1
      DO1I=1,NC
    1 H(I)=A(I)
      ZZ=0.
      DO 90 I=2,NC
      ZZ=DABS(H(I))+ZZ
   90 CONTINUE
      IF(ZZ.LT.1.D-10) GO TO 100
      P=0.
      Q=0.
      R=0.
    3 IF(H(1))4,2,4
    2 NC=NC-1
      V(NC)=0.
      U(NC)=0.
      DO1002I=1,NC
 1002 H(I)=H(I+1)
      GOTO3
    4 IF(NC-1)5,100,5
    5 IF(NC-2)7,6,7
    6 R=-H(1)/H(2)
      GOTO50
    7 IF(NC-3)9,8,9
    8 P=H(2)/H(3)
      Q=H(1)/H(3)
      GOTO70
    9 IF(DABS(H(NC-1)/H(NC))-DABS(H(2)/H(1)))10,19,19
   10 IREV=-IREV
      M=NC/2
      DO11I=1,M
      NL=NC+1-I
      F=H(NL)
      H(NL)=H(I)
   11 H(I)=F
      IF(Q)13,12,13
   12 P=0.
      GOTO15
   13 P=P/Q
      Q=1./Q
   15 IF(R)16,19,16
   16 R=1./R
   19 E=5.D-10
      B(NC)=H(NC)
      C(NC)=H(NC)
      B(NC+1)=0.
      C(NC+1)=0.
      NP=NC-1
   20 DO49J=1,1000
      DO21I1=1,NP
      I=NC-I1
      B(I)=H(I)+R*B(I+1)
   21 C(I)=B(I)+R*C(I+1)
      IF(DABS(B(1)/H(1))-E)50,50,24
   24 IF(C(2))23,22,23
   22 R=R+1.
      GOTO30
   23 R=R-B(1)/C(2)
   30 DO37I1=1,NP
      I=NC-I1
      B(I)=H(I)-P*B(I+1)-Q*B(I+2)
   37 C(I)=B(I)-P*C(I+1)-Q*C(I+2)
      IF(H(2))32,31,32
   31 IF(DABS(B(2)/H(1))-E)33,33,34
```

```
 32 IF(DABS(B(2)/H(2))-E)33,33,34
 33 IF(DABS(B(1)/H(1))-E)70,70,34
 34 CBAR=C(2)-B(2)
    D=C(3)**2-CBAR*C(4)
    IF(D)36,35,36
 35 P=P-2.
    Q=Q*(Q+1.)
    GOTO49
 36 P=P+(B(2)*C(3)-B(1)*C(4))/D
    Q=Q+(-B(2)*CBAR+B(1)*C(3))/D
 49 CONTINUE
    E=E*10.
    GOTO20
 50 NC=NC-1
    V(NC)=0.
    IF(IREV)51,52,52
 51 U(NC)=1./R
    GOTO53
 52 U(NC)=R
 53 DO54I=1,NC
 54 H(I)=B(I+1)
    GOTO4
 70 NC=NC-2
    IF(IREV)71,72,72
 71 QP=1./Q
    PP=P/(Q*2.0)
    GOTO73
 72 QP=Q
    PP=P/2.0
 73 F=(PP)**2-QP
    IF(F)74,75,75
 74 U(NC+1)=-PP
    U(NC)=-PP
    V(NC+1)=DSQRT(-F)
    V(NC)=-V(NC+1)
    GOTO76
 75 IF(PP)81,80,81
 80 U(NC+1)=-DSQRT(F)
    GO TO 82
 81 U(NC+1)=-(PP/DABS(PP))*(DABS(PP)+DSQRT(F))
 82 CONTINUE
    V(NC+1)=0.
    U(NC)=QP/U(NC+1)
    V(NC)=0.
 76 DO77I=1,NC
 77 H(I)=B(I+2)
    GOTO4
100 RETURN
     END
```

```
      FUNCTION DET(A,KC)
C     THIS FUNCTION SUBPROGRAM FINDS THE DETERMINANT OF A MATRIX
C     USING DIAGONALIZATION PROCEDURE
      DOUBLE PRECISION A(10,10),B(10,10),TEMP,DET
      IREV =0
      DO1 I=1,KC
      DO1 J=1,KC
    1 B(I,J)=A(I,J)
      DO20I=1,KC
      K=I
    9 IF(B(K,I))10,11,10
   11 K=K+1
      IF(K-KC) 9,9,51
   10 IF(I-K) 12,14,51
   12 DO13M=1,KC
      TEMP=B(I,M)
      B(I,M)=B(K,M)
   13 B(K,M)=TEMP
      IREV = IREV+1
   14 II=I+1
      IF(II.GT.KC) GO TO 20
      DO17 M=II,KC
   18 IF(B(M,I)) 19,17,19
   19 TEMP =B(M,I)/B(I,I)
      DO16N=I,KC
   16 B(M,N)=B(M,N)-B(I,N)*TEMP
   17 CONTINUE
   20 CONTINUE
      DET=1.
      DO2 I=1,KC
    2 DET=DET*B(I,I)
      DET=(-1.)**IREV*DET
      RETURN
   51 DET=0
      RETURN
      END
```

```
      SUBROUTINE MPY(B,G,N,ANS,NS)
      DOUBLE PRECISION B(10),G(10),W(10,10),ANS(11),ZED(10,10,10)
C   FORMS A SCALAR POLYNOMIAL FROM A MATRIX POLYNOMIAL
      COMMON ZED
      DO 1  I=1,N
      DO 1  J=1,N
      W(J,I)=0.0
      DO 1  K=1,N
    1 W(J,I)=W(J,I)+ZED(I,J,K)*B(K)
      DO 2  I=1,N
      ANS(I)=0.0
      DO 2  J=1,N
    2 ANS(I)=ANS(I)+W(J,I)*G(J)
      NN=N+1
      ANS(NN)=0.0
      NS=N-1
      RETURN
      END
```

```
      SUBROUTINE CHREQ(A,N,C,NRM)
C  THIS SUBROUTINE FINDS THE COEFFICIENTS OF THE CHARACTERISTIC
C     POLYNOMIAL USING THE LEVERRIER ALGORITHM
      DOUBLE PRECISION A(10,10),C(11),ATEMP(10,10),PROD(10,10),
     1                 ZED(10,10,10)
      COMMON ZED
      DATA ATEMP/100*0.0/
 1000 FORMAT (1H0,5X,31HTHE MATRIX COEFFICIENTS OF THE    ,
     *    33HNUMERATOR OF THE RESOLVENT MATRIX  )
 1001 FORMAT (1H0,  5X,29HTHE MATRIX COEFFICIENT OF S**,I1/)
 1002 FORMAT (1P6E20.7)
 1003 FORMAT (1H0,45(1H*))
      CALL CHREQA(A,N,C)
      DO 65 I=1,N
   65 ATEMP(I,I)=1.0
   70 DO 80 I=1,N
      DO 80 J=1,N
   80 ZED(N,I,J)=ATEMP(I,J)
      IF (NRM.NE.0) GO TO 71
      WRITE (6,1003)
      WRITE (6,1000)
      M=N-1
      WRITE (6,1001) M
      DO 35 I=1,N
   35 WRITE (6,1002) (ATEMP(I,J),J=1,N)
   71 DO 40 I=1,N
      DO 40 J=1,N
   40 ATEMP(I,J)=A(I,J)
      DO 10 I=1,N
      NNN=N-I
      IF (I.EQ.1) GO TO 55
      IF (NRM.NE.0) GO TO 60
      WRITE (6,1001) NNN
      DO 45 J=1,N
   45 WRITE (6,1002) (ATEMP(J,K),K=1,N)
   60 NP=NNN+1
      DO 90 II=1,N
      DO 90 J=1,N
   90 ZED(NP,II,J)=ATEMP(II,J)
      DO 15 J=1,N
      DO 15 K=1,N
      PROD(J,K)=0.0
      DO 15 L=1,N
   15 PROD(J,K)=PROD(J,K)+(A(J,L)*ATEMP(L,K))
      DO 13 J=1,N
      DO 13 K=1,N
   13 ATEMP(J,K)=PROD(J,K)
   55 DO 10 J=1,N
      NZ=N-I+1
   10 ATEMP(J,J)=ATEMP(J,J)+C(NZ)
      RETURN
      END
```

```
      SUBROUTINE CHREQA(A,N,C)
      DOUBLE PRECISION C(11),B(10,10),A(10,10),D(300)
      DIMENSION J(11)
      NN=N+1
      DO 20 I=1,NN
   20 C(I)=0.0
      C(NN)=1.0
      DO 14 M=1,N
      K=0
      L=1
      J(1)=1
      GO TO 2
    1 J(L)=J(L)+1
    2 IF(L-M) 3,5,50
    3 MM=M-1
      DO 4 I=L,MM
      II=I+1
    4 J(II)=J(I)+1
    5 DO 10 I=1,M
      DO 10 KK=1,M
      NR=J(I)
      NC=J(KK)
   10 B(I,KK)=A(NR,NC)
      K=K+1
      D(K)=DET(B,M)
      DO 6 I=1,M
      L=M-I+1
      IF(J(L)-(N-M+L)) 1,6,50
    6 CONTINUE
      M1=N-M+1
      DO 14 I=1,K
   14 C(M1)=C(M1)+D(I)*(-1.0)**M
      RETURN
   50 PRINT 2000
 2000 FORMAT (1H0,5X,15HERROR IN CHREQA)
      RETURN
      END
```

```
      SUBROUTINE DIMCH(A,NA,B)
      DOUBLE PRECISION A(10,10),B(10,10)
      DIMENSION NA(2)
      II=NA(1)
      JJ=NA(2)
      DO 1 I=1,II
      DO 1 J=1,JJ
      JARG=(J-1)*NA(1)+I
      B(I,J)=A(JARG,1)
1     CONTINUE
      RETURN
      END
```

DISTRIBUTION LIST

		No. of Copies
1.	Library Code 0212 Naval Postgraduate School Monterey, CA 93940	2
2.	Defense Documentation Center Cameron Station Alexandria, VA 22314	2
3.	Department of Aeronautics Code 57 Naval Postgraduate School	
	Prof. R. W. Bell	1
	Asst. Prof. R. A. Hess	20
	Monterey, CA 93940	
4.	Dean of Research Code 023 Naval Postgraduate School Monterey, CA 93940	1
5.	Guidance and Navigation Branch Flight Systems Research Division Ames Research Center	
	Mr. George Xenakis	1
	Mr. Leonard McGee	1
	Moffett Field, CA 94035	